1 Contents

- A One-Occupation Model 2
- Mapping into HS2013 Framework 3

This lecture is yet another part of a suite of lectures that use the quantecon DLE class to instantiate models within the [1] class of models described in detail in Recursive Models of Dynamic Linear Economies.

In addition to what’s included in Anaconda, this lecture uses the quantecon library

In [1]: !pip install --upgrade quantecon

We’ll also need the following imports:

In [2]: import numpy as np
import matplotlib.pyplot as plt
from quantecon import LQ
from collections import namedtuple
from quantecon import DLE
from math import sqrt
%matplotlib inline

2 A One-Occupation Model


- a stock of “Engineers” $N_t$
- a number of new entrants in engineering school, $n_t$
- the wage rate of engineers, $w_t$

It takes $k$ periods of schooling to become an engineer.

The model consists of the following equations:

- a demand curve for engineers:
\[ w_t = -\alpha_d N_t + \epsilon_{dt} \]

- a time-to-build structure of the education process:

\[ N_{t+k} = \delta N_{t+k-1} + n_t \]

- a definition of the discounted present value of each new engineering student:

\[ v_t = \beta k \sum_{j=0}^{\infty} (\beta \delta_N)^j w_{t+k+j} \]

- a supply curve of new students driven by present value \( v_t \):

\[ n_t = \alpha_s v_t + \epsilon_{st} \]

### 3 Mapping into HS2013 Framework

We represent this model in the [1] framework by

- sweeping the time-to-build structure and the demand for engineers into the household technology, and
- putting the supply of engineers into the technology for producing goods

#### 3.1 Preferences

\[ \Pi = 0, \Lambda = \begin{bmatrix} \alpha_d & 0 & \cdots & 0 \end{bmatrix}, \Delta_h = \begin{bmatrix} \delta_N & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \Theta_h = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ 1 \end{bmatrix} \]

where \( \Lambda \) is a \( k+1 \times 1 \) matrix, \( \Delta_h \) is a \( k \times k+1 \) matrix, and \( \Theta_h \) is a \( k+1 \times 1 \) matrix.

This specification sets \( N_t = h_{1t-1}, n_t = c_t, h_{\tau+1,t-1} = n_{t-(k-\tau)} \) for \( \tau = 1, \ldots, k \).

Below we set things up so that the number of years of education, \( k \), can be varied.

#### 3.2 Technology

To capture Ryoo and Rosen’s [2] supply curve, we use the physical technology:

\[ c_t = i_t + d_{1t} \]

\[ \psi_1 i_t = g_t \]

where \( \psi_1 \) is inversely proportional to \( \alpha_s \).
3.3 Information

Because we want $b_t = \epsilon_{dt}$ and $d_{1t} = \epsilon_{st}$, we set

$$A_{22} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \rho_s & 0 \\ 0 & 0 & \rho_d \end{bmatrix}, \quad C_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \quad U_b = \begin{bmatrix} 30 & 1 \\ 0 & 0 \end{bmatrix}, \quad U_d = \begin{bmatrix} 10 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

where $\rho_s$ and $\rho_d$ describe the persistence of the supply and demand shocks.

In [3]: Information = namedtuple('Information', ['a22', 'c2', 'ub', 'ud'])
   Technology = namedtuple('Technology', ['ϕ_c', 'ϕ_g', 'ϕ_i', 'γ', 'δ_k', 'θ_k'])
   Preferences = namedtuple('Preferences', ['β', 'l_λ', 'π_h', 'δ_h', 'θ_h'])

3.4 Effects of Changes in Education Technology and Demand

We now study how changing

- the number of years of education required to become an engineer and
- the slope of the demand curve

affects responses to demand shocks.

To begin, we set $k = 4$ and $\alpha_d = 0.1$

In [4]: k = 4  # Number of periods of schooling required to become an engineer
   β = np.array([[1 / 1.05]])
   α_d = np.array([[0.1]])
   α_s = 1
   ε_1 = 1e-7
   λ_1 = np.ones((1, k)) * ε_1
   # Use of ε_1 is trick to aquire detectability, see HS2013 p. 228 footnote 4
   l_λ = np.hstack((α_d, λ_1))
   π_h = np.array([[0]])
   δ_n = np.array([[0.95]])
   d1 = np.vstack((δ_n, np.zeros((k - 1, 1))))
   d2 = np.hstack((d1, np.eye(k)))
   δ_h = np.vstack((d2, np.zeros((1, k + 1))))
   θ_h = np.vstack((np.zeros((k, 1)), np.ones((1, 1))))

   ϕ_1 = 1 / α_s
   ϕ_c = np.array([[1], [0]])
   ϕ_g = np.array([[0], [-1]])
   ϕ_i = np.array([[-1], [ϕ_1]])
   γ = np.array([[0], [0]])
   δ_k = np.array([[0]])
   θ_k = np.array([[0]])
\(\rho_s = 0.8\)
\(\rho_d = 0.8\)

\[a22 = \text{np.array}([[1, 0, 0],
                         [0, \rho_s, 0],
                         [0, 0, \rho_d]])\]

\[c2 = \text{np.array}([[0, 0], [10, 0], [0, 10]])\]
\[ub = \text{np.array}([[30, 0, 1]])\]
\[ud = \text{np.array}([[10, 1, 0], [0, 0, 0]])\]

\[\text{info1} = \text{Information}(a22, c2, ub, ud)\]
\[\text{tech1} = \text{Technology}(\phi_c, \phi_g, \phi_i, \gamma, \delta_k, \theta_k)\]
\[\text{pref1} = \text{Preferences}(\beta, l_{\lambda}, \pi_h, \delta_h, \theta_h)\]
\[\text{econ1} = \text{DLE} (\text{info1}, \text{tech1}, \text{pref1})\]

We create three other instances by:

1. Raising \(\alpha_d\) to 2
2. Raising \(k\) to 7
3. Raising \(k\) to 10

**In [5]:**
\[\alpha_d = \text{np.array}([[2]])\]
\[l_{\lambda} = \text{np.hstack}((\alpha_d, \lambda_1))\]
\[\text{pref2} = \text{Preferences}(\beta, l_{\lambda}, \pi_h, \delta_h, \theta_h)\]
\[\text{econ2} = \text{DLE} (\text{info1}, \text{tech1}, \text{pref2})\]

\[\alpha_d = \text{np.array}([[0.1]])\]

\[k = 7\]
\[\lambda_1 = \text{np.ones}((1, k)) * \varepsilon_1\]
\[l_{\lambda} = \text{np.hstack}((\alpha_d, \lambda_1))\]
\[d1 = \text{np.vstack}((\delta_n, \text{np.zeros}(k - 1, 1)))\]
\[d2 = \text{np.hstack}((d1, \text{np.eye}(k)))\]
\[\delta_h = \text{np.vstack}((d2, \text{np.zeros}(1, k + 1)))\]
\[\theta_h = \text{np.vstack}((\text{np.zeros}(k, 1), \text{np.ones}((1, 1))))\]

\[\text{Pref3} = \text{Preferences}(\beta, l_{\lambda}, \pi_h, \delta_h, \theta_h)\]
\[\text{econ3} = \text{DLE} (\text{info1}, \text{tech1}, \text{Pref3})\]

\[k = 10\]
\[\lambda_1 = \text{np.ones}((1, k)) * \varepsilon_1\]
\[l_{\lambda} = \text{np.hstack}((\alpha_d, \lambda_1))\]
\[d1 = \text{np.vstack}((\delta_n, \text{np.zeros}(k - 1, 1)))\]
\[d2 = \text{np.hstack}((d1, \text{np.eye}(k)))\]
\[\delta_h = \text{np.vstack}((d2, \text{np.zeros}(1, k + 1)))\]
\[\theta_h = \text{np.vstack}((\text{np.zeros}(k, 1), \text{np.ones}((1, 1))))\]

\[\text{pref4} = \text{Preferences}(\beta, l_{\lambda}, \pi_h, \delta_h, \theta_h)\]
\[\text{econ4} = \text{DLE} (\text{info1}, \text{tech1}, \text{pref4})\]

\[\text{shock\_demand} = \text{np.array}([[0], [1]])\]
The first figure plots the impulse response of \( n_t \) (on the left) and \( N_t \) (on the right) to a positive demand shock, for \( \alpha_d = 0.1 \) and \( \alpha_d = 2 \).

When \( \alpha_d = 2 \), the number of new students \( n_t \) rises initially, but the response then turns negative.

A positive demand shock raises wages, drawing new students into the profession.

However, these new students raise \( N_t \).

The higher is \( \alpha_d \), the larger the effect of this rise in \( N_t \) on wages.

This counteracts the demand shock’s positive effect on wages, reducing the number of new students in subsequent periods.

Consequently, when \( \alpha_d \) is lower, the effect of a demand shock on \( N_t \) is larger.

The next figure plots the impulse response of \( n_t \) (on the left) and \( N_t \) (on the right) to a positive demand shock, for \( k = 4 \), \( k = 7 \) and \( k = 10 \) (with \( \alpha_d = 0.1 \)).
Both panels in the above figure show that raising $k$ lowers the effect of a positive demand shock on entry into the engineering profession.

Increasing the number of periods of schooling lowers the number of new students in response to a demand shock.

This occurs because with longer required schooling, new students ultimately benefit less from the impact of that shock on wages.

References
